[1]Design and Implementation Issues for A Distributed Access Control System

Srilekha S. Mudumbai, William Johnston, Mary Thompson, Gary Hoo, and Abdelilah Essiari

Information and Computing Sciences Division

Ernest Orlando Lawrence Berkeley National Laboratory

University of California

Abstract

Remote access to widely distributed systems comprised of scientific instruments, large-scale storage systems, and network-based multimedia collaboration channels require distributed access control to prevent unauthorized access. Akenti[1] is a distributed access control system designed to provide a flexible, easily managed mechanism, accommodating multiple owners, which strongly controls access to distributed resources by widely distributed users. In addition to access control, the capabilities of the Akenti system include certificate acquisition; caching for better wide area performance; and monitoring of the access control decision process to provide user-level feedback.

1.0 Introduction

In many environments, e.g. multi-institutional science, the people with authority to grant access to resources (stakeholders) may be both physically and organizationally remote from the resource. A simple approach to access control would be a per-resource policy defined by the stakeholder who owns the resource. But when the resource is owned by several stakeholders, an expression of the policy defined by each stakeholder is essential. An access control mechanism for resources owned by distributed stakeholders should

[1] The work described in this paper is supported by the U. S. Dept. of Energy, Office of Energy Research, Office of Computational and Technology Research, Mathematical, Information, and Computational sciences and ERLTT Divisions under contract DE-AC03-76SF00098 with the University of California, and by DARPA, Information Technology Office. This is report no. LBNL-41911, SSMudumbai@lbl.gov

provide distributed management of all of the secure information that is the basis for access control decisions.

In this paper we discuss the design and the implementation issues of the Akenti access control system. This system enables stakeholders to remotely and securely create and distribute instructions authorizing access to their resources. Akenti uses a public key infrastructure (PKI)[2,3,4] and various types of digitally signed certificates (identity, use-condition, attribute, and capability) that are maintained in Internet-based Web and Lightweight Directory Access Protocol (LDAP)[5,6] servers to accomplish the access control. Akenti provides secure, policy-based access control in an environment where

- The stakeholders of resources may be distributed
- The representation of use-conditions is applicable to a generic policy model. A typical policy model is hierarchical. Our current policy model distinguishes between stakeholder-based policies and an overall policy set by the administrators of the entire set of resources
- The stakeholders can belong to different organizations
- The resources may be distributed
- Users may access resources remotely

From the very first use of Akenti, it was obvious that it had two major challenges to overcome: the length of time it takes to gather all the certificates that are distributed on remote servers; and making and explaining the access control decisions that are required by the stakeholders. This paper will discuss in some detail the caching mechanism by which Akenti reduces certificate-gathering time, and the monitoring of access control decisions. We also discuss the possible modes of system failure, and outline both short- and long-term future plans for Akenti's development and deployment.

2.0 Background Technology

Akenti is designed to use currently available distributed security technologies. In particular it uses Secure Sockets Layer-based (SSL)[7] secure Web servers to provide

remote access to resources and remotely stored certificates; public/private key signed certificates to express user identity, resource use conditions, and user attributes; PKI certificate authorities (CA) and LDAP servers to manage the certificates.

In public/private key encryption[8] schemes, the public key can be circulated to anybody in the world while the private key is kept secret by its owner. These keys have the property that encryption by one key followed by decryption by the other key recovers the original data. This guarantees that the data encrypted by a user's public key can be recovered only by the owner of the matching private key, accomplishing data confidentiality. Alternatively, encryption by the private key followed by decryption by the public key ensures the decoder that the message came from the owner of the private key. This method is used with message digests for authentication (digital signatures) and to ensure data integrity.

The PKI is a set of standards and services that use public key cryptography and X.509[8] certificates for issuing and managing identity certificates in a networked environment. A trusted third party, known as a certificate authority (CA), provides a mapping between a person and his public key, certifying the latter by digitally signing a certificate containing the key.

The SSL protocol provides secure peer-to-peer communication. It provides privacy between a client and a server by encrypting the data sent between them. It also authenticates the server, and optionally the client; in our access control system, we require bi-directional certificate-based authentication. Akenti operates on top of the resulting encrypted channel. Akenti controls access to information on the server by matching the user's identity, as presented by the client, with other digitally signed documents (use-condition and attribute certificates) that establish an access control policy.

3.0 Akenti Design

3.1 Goals

- To allow specific remote users transparent remote access to resources and to strongly deny such access to non-authorized users.
- To allow remote stakeholders to easily enforce their requirements for access.
- To make access decisions quickly enough so that secure access times are comparable to normal remote access times.
- To make information available on why access succeeded or failed.

3.2 Terminology

- A *resource* may be information, processing or communication capabilities or physical systems (instruments such as telescopes or microscopes), etc.
- *Access* means the ability either to obtain information from the resource (as in "read" access), or to modify the resource ("write" access), or to cause that resource to perform certain operations (“execution” access). A network-based server acting as a proxy for the actual resource typically provides the mechanism for remote access.
- A *user* is an entity attempting to gain access to the resource via a client. A client has to participate in a series of authentication and verification steps based on the user’s credentials before being granted access to the resource.
- *Authentication* is the establishing of the identity of the entity with which one is communicating. In our view of access control, authentication is a two-way process. Initially, the client authenticates a server before it attempts to gain access to the resource controlled by the server. Subsequently, the server authenticates the client. *Stakeholders* own or have a proprietary interest in a resource and determine who may access it. Stakeholders may be geographically distributed.
- *Use-condition certificates* are digitally signed documents that express the criteria that a user must satisfy to access a resource.
- *Attribute certificates* are digitally signed documents that represent characteristics of users that satisfy specific use-conditions.

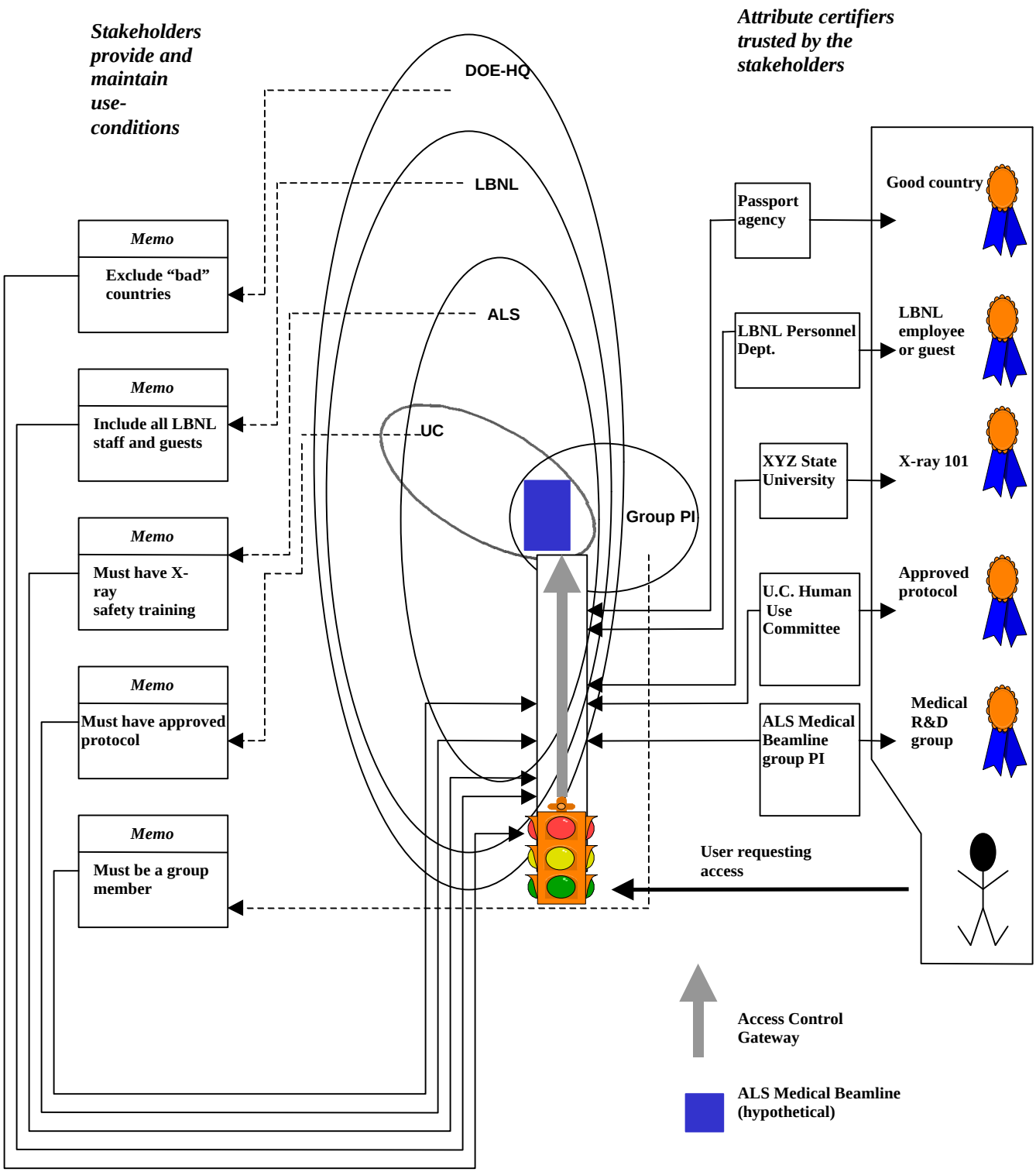

Figure1 describes a common societal access control model.

3.3 Examples of stakeholders and resource model

As an illustration of independent and distributed stakeholders asserting their authority over a commonly owned resource, we consider a scientific resource at a government-operated laboratory. This is illustrated in Figure 1[9].

Our example resource is the fictional Advanced Light Source (ALS - an ultra-high intensity X-ray source) medical beam line at the Lawrence Berkeley National Laboratory operated by the University of California for the U.S. Dept. of Energy. The ultimate stakeholder in charge of the resource is the Dept. of Energy. However, there are also other stakeholders, each of which controls access to this resource by imposing use-conditions. Some stakeholders are related to one another in a hierarchical manner: the Dept. of Energy sets broad policy for use of the National Laboratories; the Laboratory (LBNL) sets site access rules; the administration of the ALS sets the safety rules; and the principal investigator (the "owner" of this resource) establishes who participates in the experiments. Other stakeholders may not be part of the main hierarchy of stakeholders, but may nonetheless have their own compelling interest in the resource. For example, the University of California has management oversight of the Laboratory and, among other things, must review and agree to all treatment protocols involving humans. Each of these stakeholder use-conditions has one or more attribute certifiers, as indicated on the right side of Figure 1. The access control gateway forbids or allows access to resource based on matching the attribute certificates held by the user with those required by the use-conditions. The abstraction of this societal model is what we wish to achieve our computer-based access control system.

3.4 Akenti Infrastructure

Akenti depends on a number of different components, some of which are commercial products and some of which are unique to Akenti.

The ***Certification Authority Server*** issues and manages public key/identity certificates. This includes maintaining a directory service, and renewing and revoking certificates. We use the Netscape Certificate Server[10] for this purpose. A web browser acts as a client on behalf of the user and forwards requests for signing identity certificates to the CA.

A ***Directory Server*** manages identity certificates. A Lightweight Directory Access Protocol (LDAP) server provides a flexible way of managing and accessing certificates within an organization. The CA can register certificates for any identities that have LDAP entries. These certificates are then accessible via the Internet.

A ***Database Server*** can be used to store identity certificates and can be queried to obtain certificates in case a directory server is not available. The database server should be reachable via a Common Gateway Interface.

A ***Web Certificate Server*** stores use-condition and attribute certificates. These can be fetched as via a URL.

A ***Web Server Resource Gateway*** acts as the security gateway for remote access to resources. Akenti uses an SSL-enhanced Apache[11] server for this purpose.

The Netscape browser provides ***Key Pair Generation and Management*** for user identities. However, the keys that are used by stakeholders to sign use-condition certificates and by attribute issuers to sign attribute certificates must be available outside of a browser. Hence we also provide a small Java application to generate and store keys. The public key/identity thus generated must be submitted to a CA for certification and entered into the CA's LDAP database.

Java ***Certificate Generation applications*** are distributed to various stakeholders and attribute certifiers to generate use-condition and attribute certificates. The stakeholders and attribute certifiers can sign the certificates only after their identities are authorized by the CA; otherwise, the certificates are considered invalid.

4.0 Akenti System Architecture

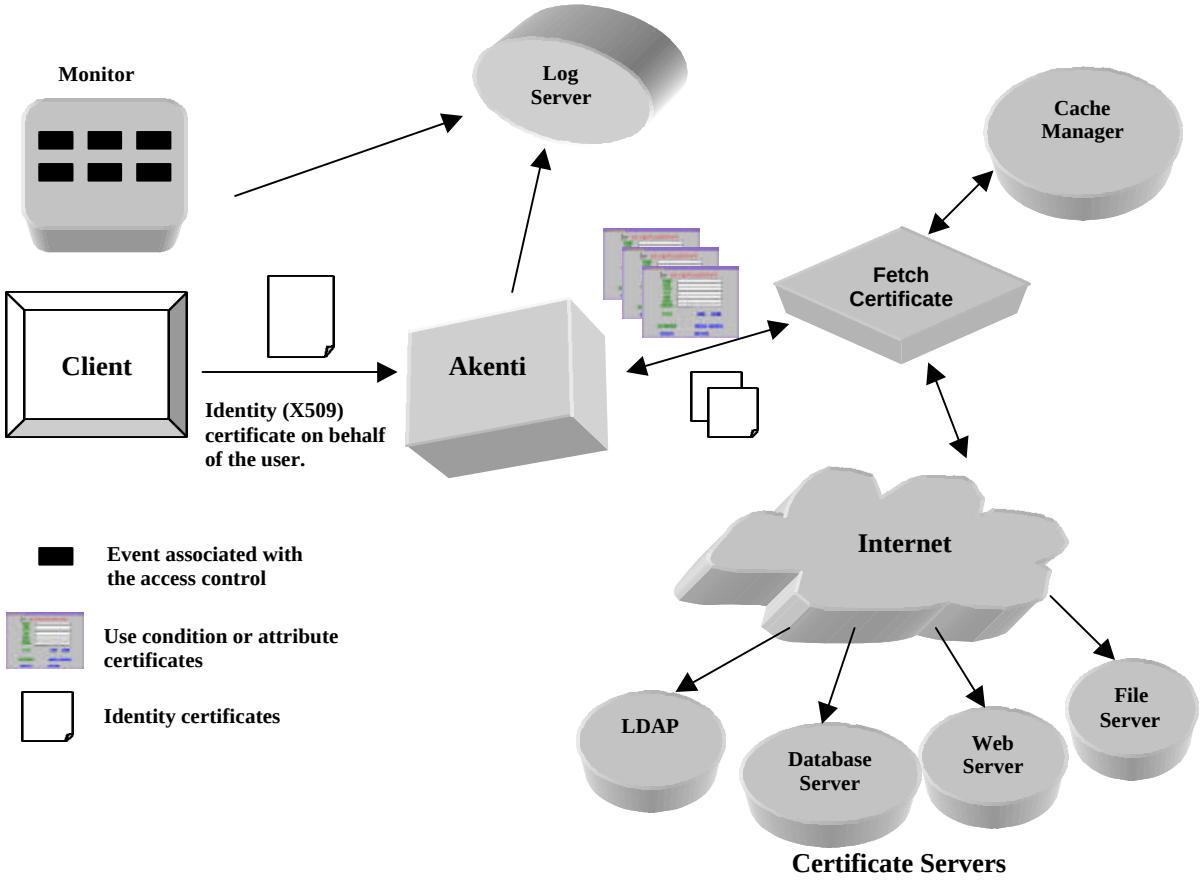

Figure 2: Illustration of Akenti architecture.

In Akenti's client-server architecture, clients (on behalf of users) attempt to access resources that are controlled by servers.

A priori authorization requirements govern which clients may access which servers for what purposes and under what conditions. These requirements are reflected in digitally signed use-condition certificates. A particular entity's satisfaction of a requirement is reflected in a digitally signed attribute certificate. Akenti fetches all use-condition certificates applicable to a resource and attempts to locate a set of attribute certificates that fulfils the union of all the use-conditions. Both use-condition and attribute certificates will usually be widely distributed across multiple hosts. Existing PKI and

security systems provide confidentiality, message integrity, and user identity authentication, during and after the access decision process. A Certificate cache server is used to store certificates for a short time after they have been acquired over the Internet: Akenti contacts the cache server before it performs wide-area searches for certificates. A log server captures Akenti events during the access control process. A Java application at the client side can listen to the log server and can display the access control process for a session as it happens. (Fig. 2)

4.1 Components of the Access Control Model

A ***Policy file*** is securely associated with each resource. This is relatively static information that specifies who can provide access information (i.e., defines the stakeholders), where to look for access control information, explicitly names all the trusted CA's and their public key.

Use-condition certificates are digitally signed documents, created and manipulated perhaps remotely by resource stakeholders, that specify the conditions that must be met by a user desiring access to a resource. They include

- Combinations of required attributes and values
- Validity period
- Name of the resource
- Access and allowed actions
- Authorities trusted to issue attribute certificates
- CA to be trusted to verify the user and issuer identities
- Signature of the use-condition issuer.

Attribute certificates are digitally signed documents, stored in trusted servers, that certify that a user possesses a specific attribute (for example, membership in a named group, has completed a certain training course, etc.). They include

- Attribute name
- Value

- Validity period
- Auxiliary information (e.g. time interval)
- Subject (User) and its trusted CA
- Issuer and its trusted CA
- Signature of the attribute certificate issuer.

Identity (X.509) certificates associate an entity's name with a public key and bind them together through the digital signature of a CA. (The corresponding private key is held secret by the user, and is used to prove "ownership" of the corresponding public key). Any user desiring to gain access to a resource must present a valid identity certificate. The CA that signed the certificates must be trusted by the owner of the resources.

4.2 Authentication and Access

In the first phase of authorization, the client authenticates the server and the server authenticates the client. When a client contacts a secure server, it is presented with the server's identity certificate. If the client trusts the CA that signed the server's certificate, the client presents the user's identity in the form of an X.509 certificate; otherwise, it cancels the session. The initial authorization succeeds if the user's identity certificate was issued by a CA trusted by the server. The SSL protocol is responsible for this initial phase of authorization.

The second phase of authorization is performed by Akenti. It re-fetches the user's identity certificate from the LDAP server associated with the CA that issued the certificate. This is an attempt to ensure that the certificate has not been revoked. It also verifies the CA's signature against the CA's public key, stored in a trusted place. This is a stronger check than is normally done by the SSL libraries.

4.3 Search Mechanism for Use Conditions and attributes

There may be multiple stakeholders physically and organizationally remote who impose use-conditions on a resource. In the policy file each stakeholder must specify a location where all the use-conditions imposed by that stakeholder are kept. This may be in the form of a single URL if there is just one use-condition certificate, or a URL to a directory in which multiple use-conditions are kept. A local directory pathname may also be specified. In the future we plan to add LDAP search directories as well. Akenti must be able to find at least one use-condition for each stakeholder, or access to the resource will be denied. This is to prevent unauthorized access to a resource if the server on which the stakeholder's use-conditions are stored is unavailable. The policy file may specify mirror sites, but each site must have a complete set of the use-conditions.

Since Akenti must potentially search several remote directories, this raises the issue of efficient searching. The certificate generator applications generate a hash code from the searchable fields of the certificate which is used to name the certificate. The Akenti search process generates the same hash code from the search query. For example, for use-conditions, the resource name is hashed. Thus Akenti can request certificates by hash name and get only those that refer to the requested resource. Attribute certificates are searched in the same manner, but the hash codes are based on the tuple <*attribute, value, issuer, issuer's CA*>, which is unique for a given attribute certificate. This restricted search is time effective.

In spite of this efficient searching, it is still time-consuming to search for and to retrieve certificates over the Internet. The next obvious step is to cache certificates locally once they have been found. Our initial use of Akenti is as a replacement for the Web's standard access control (i.e. domain and/or user-password authentication). Because the Web server can turn one logical document access into several actual accesses, certificate caching is even more critical.

For a typical Web access of the form https://host/foo, where 'foo' is a directory, the Apache requests an Akenti access check at least three times before the browser presents the requested document to the user.

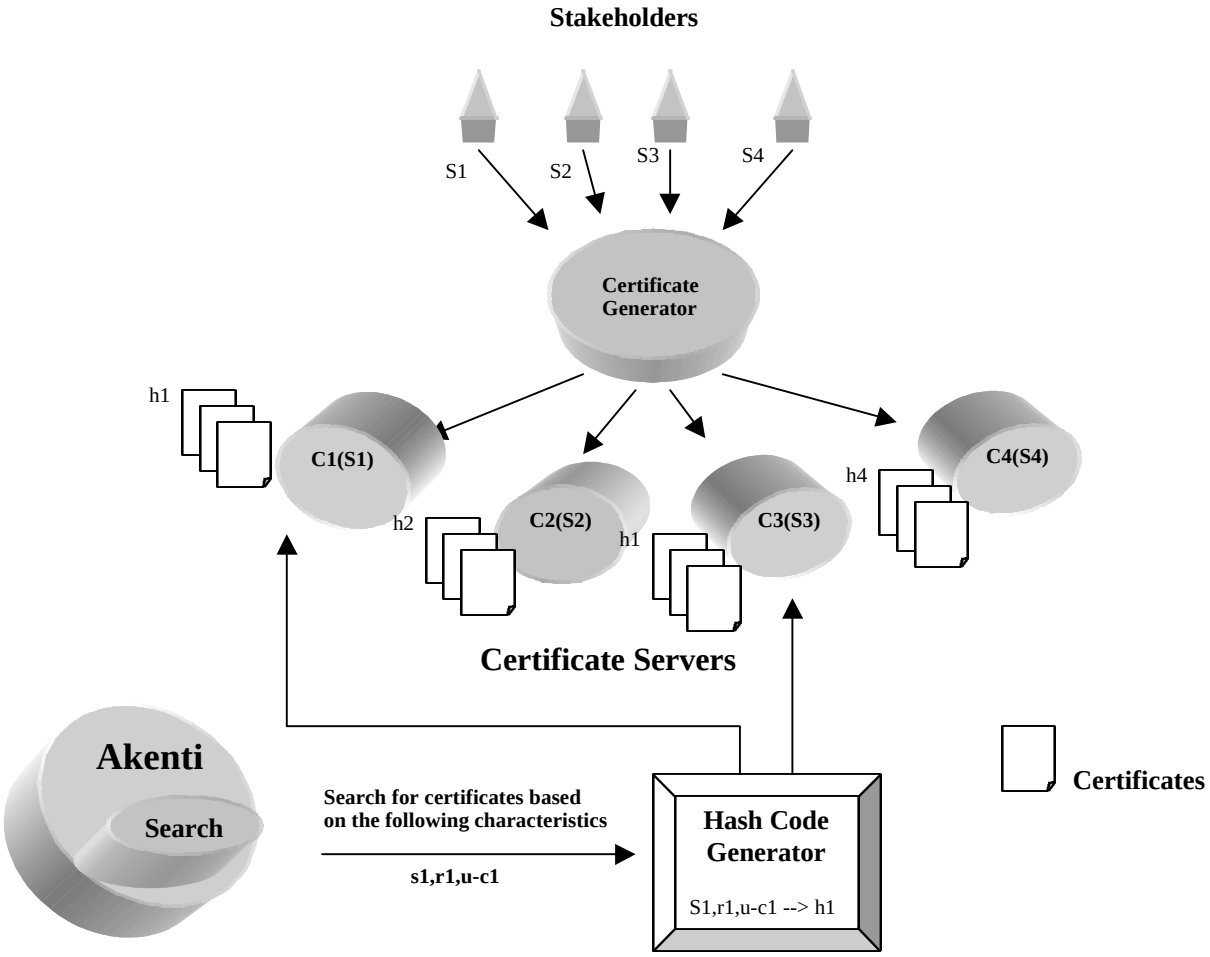

Figure 3: This figure describes searching mechanism.

- The first request is for the resource entered by the client -- https://host/foo
- The second request is made after the browser corrects its request to the directory -- https://host/foo/
- The third request is for the default file name of the directory index -- https://host/foo/index.html
- In addition for each in-line image an additional request will be made.

Because Akenti searches for certificates distributed over the Internet for each request, the overall access for the resource is considerably delayed. These multiple requests were the motivation for immediately adding caching to the Akenti system. Thus Akenti caches

required certificates when they are initially acquired. For subsequent sessions, all required certificates can be obtained from the local cache, speeding the searching process.

Another feature that became immediately necessary was some way for system developers, stakeholders and legitimate users of the resources to understand what was going on during the access control decision process. Since there can be a large number of independent use-condition and attribute certificates involved in making an access decision, we have used a logging facility to keep track of each use-condition that needs to be satisfied and each attribute certificate as it is searched for. The log reveals what certificates a user is missing if access is denied or if an unexpected access is granted, the stakeholder can determine from the log what use-condition is missing.

5.0 Caching Mechanism

The main issues of the cache design are

- what needs to be cached
- what form the cached information should take
- how certificates can be stored
- how efficiently they can be searched and accessed.

The previous section showed how certificates are obtained from the Internet in the absence of a cache. When Akenti reads a policy file for a resource, it gets information about which certificate servers (named servers) have the certificates (use-condition and attribute) that are related to the resource in question. Searching is based on hash codes computed for the resource. Cached certificates are stored by the same hash-coded filenames.

5.1 How does the caching work?

We have implemented caching by creating a Cache Manager process. The Akenti policy engine is implemented as a set of library routines that can be called by different

processes. In the case of our Akenti/Apache server several processes may be checking access for the same set of resources in parallel. We wish to share caching information among all these processes.

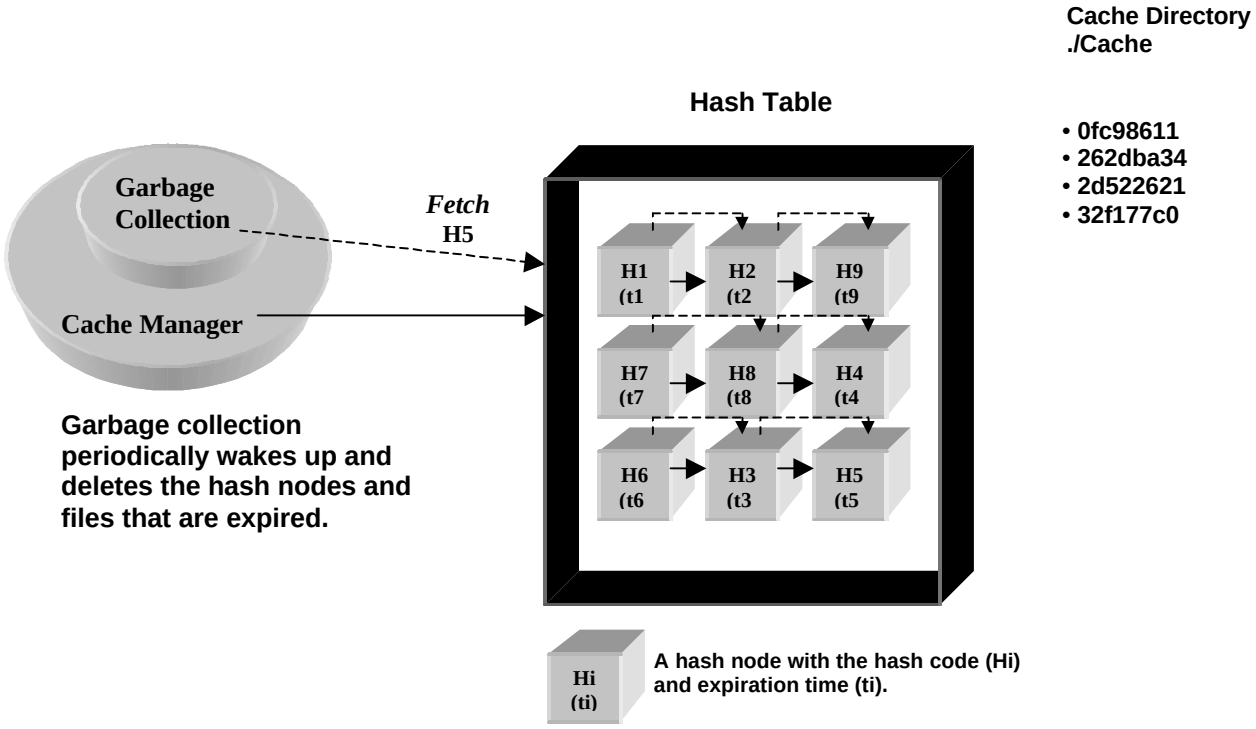

Figure 4 Cache Manager

The Cache Manager has the task of storing, searching for and retrieving cached certificates. The only thing it knows about the certificate is its name and validity period. The Cache Manager receives a 'fetch' request for a particular hash code. It searches for the name in its hash table and if it locates a matching certificate, it checks that it has not expired and returns it. If it cannot locate the hash node, returns a 'NOT FOUND' message and waits for Akenti's subsequent input of the certificate to be cached.

There is also a garbage collection facility. It runs at regular intervals to remove expired cached certificates and their hash nodes. The stakeholder can set the validity period of a cached certificate.

5.2 Contents of a Caching Certificate

The caching certificate is slightly more complicated than its current use might suggest, since in the future we may want to put the Cache Manager (CM) on a separate machine from the one the Akenti policy engine is running on.

A caching certificate, consists of a version identifier, unique object identifier (OID), validity period and the certificate (use-condition, attribute or identity) itself. If the CM runs on a host remote from the secure server, the cached certificates must be signed by the secure server.

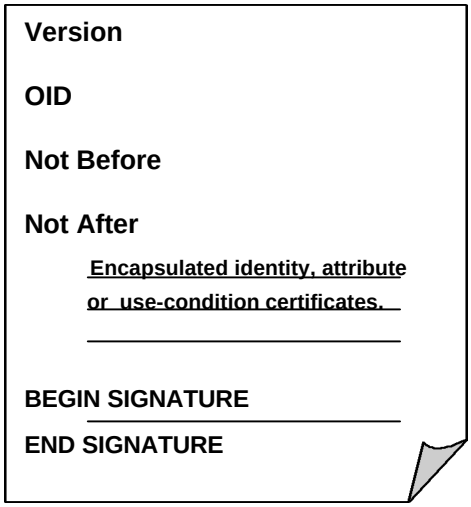

Figure 5 shows the contents of a cache certificate.

These certificates have a lifetime of a few days unless the stakeholder lowers the time. Use-conditions are cached in a slightly different format than attribute and identity certificates. A set of use-conditions for a particular resource is stored in a single cached certificate, whereas each attribute or identity certificate is stored in a separate certificate.

5.3 Capability Certificate

We have also experimented with caching a complete capability for a given user and resource. This experiment was prompted by the Web browser behavior of making several independent access requests for each logical document request. A capability consists of

user identification, resource identification, permitted action, and validity period. At present, capabilities are simply stored with the resources rather than in the Cache Manager's database. Since these capabilities are the result of many different certificates the lifetimes are kept very short (less than 5 minutes).

When Akenti performs an access control check, it authenticates the user and then checks whether there exists an unexpired capability for the user with respect to the requested resource. If so, then it simply allows access to the user with the permitted actions.

We have run an Apache server with Akenti access control, serving a set of Web pages, in three modes: without any caching, with capability caching and with certificate caching. The access times without caching are 10 to 15 times longer than insecure Web accesses and are quite variable, depending on the number of certificates that are required by the resources and the response time of all the remote servers. With capability caching, the access time for the first session (lacking a cached capability) is just as slow, but subsequent sessions revert to simple Web server access times. Since even the first session times are effectively amortized over the sessions that comprise a logical connection, the access to the resource seems sluggish, but tolerable. Certificate caching tends to reduce the Akenti overhead on an access that required nine certificates to an average time of about 1 second. This overhead, in addition to the time required for Web access, is well within the normal variation of remote accesses.

6.0 Monitoring Access Control

The assertion of use-conditions imposed by different stakeholders at different levels of a resource tree involves many independently generated certificates. We have developed tools that create these certificates with correct syntax and do some checking for logical correctness. But the interactions between the certificates can only be observed when some user attempts to access a resource.

The acquisition and processing of certificates is a complex process that is, if successful, transparent to the user, and if unsuccessful, opaque. Monitoring is important for these reasons:

- proving to a client who is granted access (or to a stakeholder) that there is actually an access control system operating
- providing sufficient information to potentially valid users who are denied access about why they were denied, i.e., what certificates they lack for this resource debugging the system in a widely distributed environment

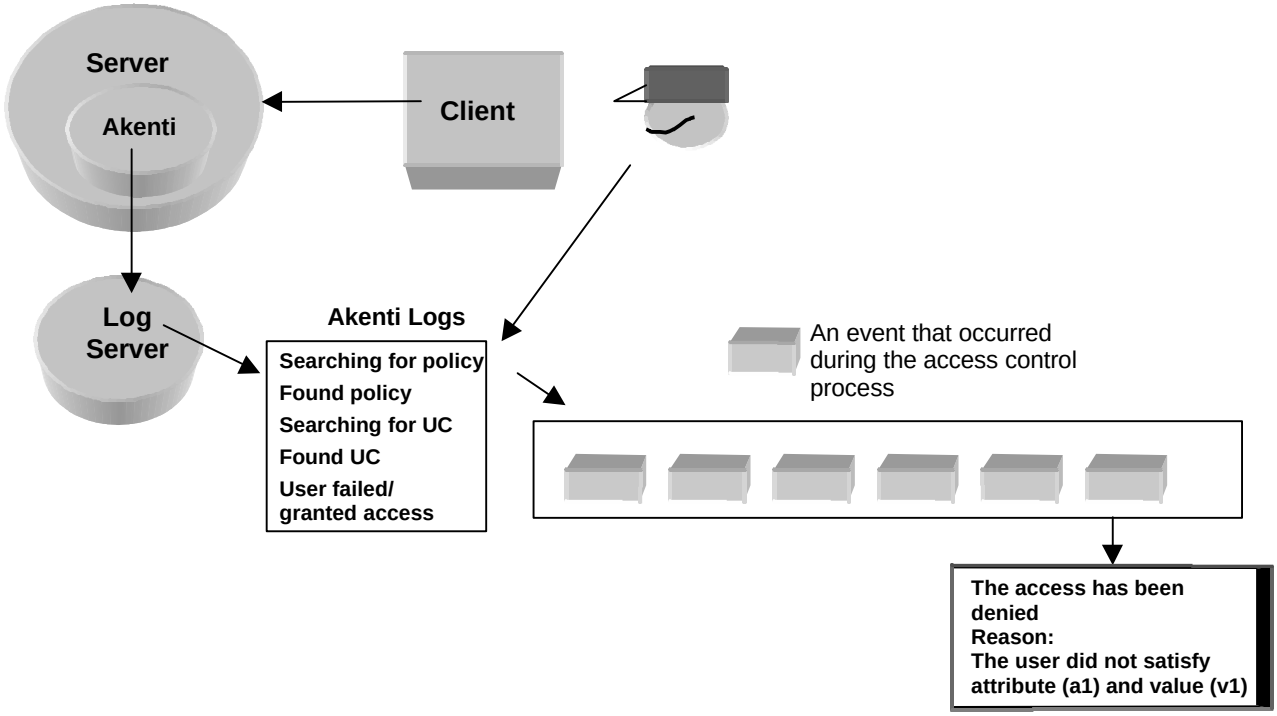

Figure 7 shows monitoring architecture

Akenti supports monitoring the access control process as it occurs, in order to understand the overall access constraints on a resource and to facilitate correcting any mistakes in the required certificates.

Currently Akenti uses an existing logging package (NetLogger[12]) to write access events in an IETF proposed standard, Universal Logger Message format to either a file, Unix *stderr* or a host and port on which a log server is listening. There are events defined for

the beginning and end of each phase of the access decision, e.g., searching a policy file to determine the stakeholders, searching for certificates, validating a certificate, etc. We are currently implementing a Java applet, which can be started during the initiation of a secure session to monitor the events as they happen. If access is denied, the applet will allow the client to know what prevented it from accessing the resource. Then the client can send requests to the appropriate stakeholders and attribute certifiers requesting that the user be granted any missing certificates to which he is entitled. Such detailed information will only be provided to a client whose identity can be verified.

A complete copy of the log file is also kept on the Akenti server machine. This provides a complete account of all accesses to a secure resource tree. We plan to build tools to digest this log into a more compact audit trail. The existing NetLogger analysis tools can be employed to give us a picture of the system's performance.

7.0 System Vulnerabilities

Any system claiming to be secure should be analyzed for likely ways in which it can fail. In Akenti we have noted two failure modes:

- If either the use-condition certificates for a desired resource or the attribute certificates for a user are not available from the named servers, Akenti forbids access. This is a failsafe decision that leads to denial of access to legitimate users in preference to allowing unauthorized access as the result of missing use-conditions. The certificates may be unavailable either because the host is unreachable on the network, or because someone has compromised the security of the certificate host and removed valid certificates. Stakeholders and attribute certifiers are responsible for making their certificates reachable via the Internet on reasonably secure hosts. To help deal with machines or network links that may be down, we allow stakeholders to name mirrored servers in addition to the original server.
- If revoked certificates are not removed from the CA's directory service, they are considered to be valid by Akenti. In the future there will likely be a standard way to

get a list of revoked certificates from a CA, and Akenti can use this to check the validity of a certificate instead of just re-fetching from the CA's directory service. At the moment, the Netscape CA automatically removes certificates from its LDAP server as they are revoked, and this serves our purpose.

8.0 Deployment of Akenti

An early deployment of Akenti is being used in support of the DOE2000 Diesel Engine Collaboratory. Akenti is incorporated in a secure Apache Web server protecting Web-based resources from unauthorized access. The web site for additional information is

http://www-itg.lbl.gov/DieselCollab

Akenti is also integrated with SPKM/GSS[13] (Simple Public-Key Mechanism/Generic Security Service API). GSS provides client-server developers with a secure message-based mechanism. Akenti's policy engine is incorporated into SPKM/GSS in order to provide access control as discussed in the above sections.

9.0 Future Work

The prototype Akenti system suggests several avenues for future work.

The performance monitoring system gathers a great deal of raw data, but lacks automated analysis tools. We envision that two types of applications will find the monitoring logs of use: interpreters to display the progress of the access decision making process to the user, and auditing tools for stakeholders and system administrators. Such analysis tools initially would be run locally, but eventually could incorporate mobile agents to monitor and to maintain access control remotely (and, of course, securely). Such tools likely would be customizable and extensible using the available new techniques in Java.

We are currently working to integrate capability certificates into the Cache Manager. Determining the appropriate certificate lifetimes is the primary challenge for the integration.

In addition to refining Akenti itself, we are working to deploy the system in different environments to gain further insights into usability issues. In particular, we plan to use Akenti as part of a distributed system scheduler; to make Akenti available for access control in a CORBA environment[14,15], to incorporate it into the collaboratory environment; and to integrate it with electronic notebooks.

10.0 Conclusion

In this paper, we have described Akenti, a distributed access control system designed to protect sensitive shared resources from unauthorized access. In designing Akenti, we had to address the challenge of time delays introduced by the distributed nature of the access control certificates. This was done through certificate caching. Also, since the access control decision is based on policy set by multiple stakeholders, there needed to be a clear way of tracking these decisions. This motivated a logging mechanism that aids in monitoring access control events during a secure session. A prototype of Akenti has been used in different scientific environments to control access to variety of resources.

For more information, a detailed view of the project can be obtained from the following web sites

http://www-itg.lbl.gov/security
http://www-itg.lbl.gov/security/Akenti

11.0 References

[1] "*The Akenti Approach*", http://www-itg.lbl.gov/security/Akenti

[2] "*Generic Security Application Program Interface*", John Linn, Sep 1993. Available at http://ds.internic.net/rfc/frc1508.txt.

Also see more recent and related drafts at the IETF Common Authentication Technology home page (http://www.ietf.cnri.reston.va.us/html.charters/cat-charter.html) and at http://www.ietf.cnri.reston.va.us/ids.by.wg/cat.html.

[3] "*Applied Cryptography"*, Second Edition. B. Schneier, John Wiley & Sons, 1996.

[4] "*Introduction to Public-Key Cryptography*" http://devedge.netscape.com/docs/manuals/security/pkin/index.html

[5] "*Netscape Directory Server*" http://home.netscape.com/directory/v3.0/index.html

[6] "*Lightweight Directory Access Protocol,*" Yeong, W., Howes, T., and S. Kille, RFC 1777, March 1995

[7] "The SSL Protocol" http://live.netscape.com/newsref/std/SSL.html

[8] "*Computer Communications Security: Principles, Standards, Protocols, and Techniques*". W. Ford, Prentice-Hall, Englewood Cliffs, New Jersey, 07632,1995.

[9] "*Authorization and Attribute Certificates for Widely Distributed Access Control*", William Johnston, Srilekha Mudumbai, and Mary Thompson, published in the Proceedings of IEEE High Performance Distributed Computing (HPDC-7)'98, 28-31, July 1998 at Chicago, Illinois.

[10] "Netscape Certificate Server" http://home.netscape.com/certificate/v1.0/index.html

[11] "*About the Apache HTTP Server Project*" http://www.apache.org

[12] "*The NetLogger Methodology for High Performance Distributed Systems Performance Analysis"*, Brian Tierney, William Johnston, Brian Crowley, Gary Hoo, Chris Brooks, Dan Gunter, published in the Proceedings of IEEE High Performance Distributed Computing (HPDC-7)'98, 28-31, July 1998 at Chicago, Illinois.

[13] "*Internet Draft on Simple Public Key Mechanism*," C. Adams, Bell-Northern Research, June 20, 1994 http://www.cs.hut.fi/crypto/ipsec/draft-ietf-cat-spkmgss-00.txt

[14] "*Standards-Based Software Infrastructure for Collaborative Environment and Distributed Computing Applications*", D. Agarwal, I. Foster and T. Strayer, White Paper.

[15] "*The Collaboratory Interoperability Framework Common Application Programming Interface*", D. Agarwal, K. Berket, N. Narasimhan, P. Schabert, I. Foster, S. Tuecke,
http://www-itg.lbl.gov/CIF/Reports/GcommonAPI.html